

DW_fp_sincos

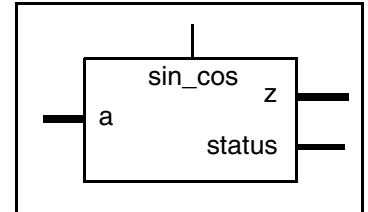
Floating-Point Sine and Cosine

Version, STAR, and myDesignWare Subscriptions: [IP Directory](#)

Features and Benefits

- The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format.
- Hardware for denormal numbers of IEEE 754 standard is selectively provided.

Revision History



Description

DW_fp_sincos is a floating-point sine and cosine unit that calculates $z = \sin(a)$, $\sin(\pi a)$, $\cos(a)$ or $\cos(\pi a)$ where a and z are floating-point values. Users can choose a function by controlling the input `sin_cos` or setting the parameter `pi_multiple`. There are two additional parameters that control the performance: `arch` is for the implementation selection between the area-optimized and the speed-optimized, and `err_range` allows users to choose between two levels of error behavior. DW_fp_sincos does not support rounding modes. The output `status` is an 8-bit optional status flag port.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	(<i>exp_width</i> + <i>sig_width</i> + 1) bits	Input	Input data (radian)
sin_cos	1 bit	Input	Function select: 0 = sine , 1 = cosine
z	(<i>exp_width</i> + <i>sig_width</i> + 1) bits	Output	Sine or cosine value
status	8 bits	Output	Status flags for the output For details, see STATUS Flags in the <i>Datapath Floating-Point Overview</i> .

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 32 bits	Word length of fraction field of floating-point numbers a and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a , z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
pi_multiple	0 or 1 Default: 1	Angle is multiplied by π : <ul style="list-style-type: none"> 0: $z = \sin(a)$ or $\cos(a)$ 1: $z = \sin(\pi a)$ or $\cos(\pi a)$
arch	0 to 3 Default: 0	Selection of optimized implementation: <ul style="list-style-type: none"> 0: Area-optimized implementation (for behavior that is backward-compatible with DC releases up to Q-2020.03) 1: Speed-optimized implementation (for behavior that is backward-compatible with DC releases up to Q-2020.03) 2: Area-optimized implementation with enhanced numerical behavior 3: Speed-optimized implementation with enhanced numerical behavior
err_range	1 or 2 Default: 1	Error range selection: <ul style="list-style-type: none"> 1: Tighter error bound 2: Looser error bound Notes: <ul style="list-style-type: none"> This parameter is ignored when <i>arch</i> = 0 or 1. Setting <i>arch</i> = 2 or 3 yields a smaller and faster circuit when <i>err_range</i> = 2 and <i>sig_width</i> > 20.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_FP_SINCOS_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_fp_sincos_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_sincos.v	Verilog simulation model source code

As shown in [Table 1-5](#), the operation of DW_fp_sincos is determined by the parameter *pi_multiple* and the input *sin_cos*. Since the sine and cosine are periodic functions with $\sin(2\pi k + \theta) = \sin(\theta)$ or $\cos(2\pi k + \theta) = \cos(\theta)$, where k is an integer, the floating-point sine/cosine operations cannot avoid the input precision error when the value of the floating-point input is larger than 2π .

Table 1-5 Operations of DW_fp_sincos

Parameter <i>pi_multiple</i>	Input <i>sin_cos</i>	Output <i>z</i>
0	0	$\sin(a)$
	1	$\cos(a)$
1	0	$\sin(\pi a)$
	1	$\cos(\pi a)$

That is, for example, if input *a* is a big number like 1.0×2^{50} for 32-bit floating-point number so that the value of the lsb bit of the significand is larger than 2π , then the input precision error becomes dominant because it is larger than the range of the angle θ . In this case, DW_fp_sincos(*a*) produces 0 when *pi_multiple* = 1 and *sin_cos* = 0, assuming $\theta = 0$. On the other hand, if *pi_multiple* = 0, *sin_cos* = 0, the calculation of the angle θ from the input *a* where $a = 2\pi k + \theta$ brings the internal truncation error. In this case, the angle θ cannot be zero and DW_fp_sincos(*a*) generates a non-zero arbitrary number. However, if the input *a* is an infinite number, DW_fp_sincos produces a NaN. Table 1-6 shows examples of DW_fp_sincos operations.

Table 1-6 Result of Special Inputs When *sig_width* = 10, *exp_width* = 5, *ieee_compliance* = 0 and *sin_cos* = 0

<i>pi_multiple</i>	Input <i>a</i>	Output <i>z</i>
0	0000_0000_0000_0000 (Zero)	0000_0000_0000_0000
1		0000_0000_0000_0000
0	0111_1100_0000_0000 (+ Infinity)	0111_1100_0000_0000
1		0111_1100_0000_0000
0	1111_1100_0000_0000 (- Infinity)	0111_1100_0000_0000
1		0111_1100_0000_0000

DW_fp_sincos provides the hardware for denormal numbers and NaNs of IEEE 754 standard. If the parameter *ieee_compliance* is turned off, denormal numbers are considered as zeros, and NaNs are considered as Infinity. Otherwise, denormal numbers and NaNs become effective and additional hardware to manipulate them is integrated.

Error Behavior

To deliver the best QoR, the DW_fp_sincos was designed so the relative error is not bounded across the entire input range. The relative error increases as the output either gets close to 0.0 or is a multiple of π . Even in the range $[-\pi, \pi]$, the relative error may be more than a few ulps when the input is near either 0.0 or $\pm \pi$.

For configurations with *arch* = 2 or 3, the approximation $\sin(a) = a$ is used when *a* is small, reducing the error in the calculation of $\sin(a)$ when the input values are near 0.0.

As the input value gets large, the computation of the equivalent reduced input angle (angles in the range of $[-\pi, \pi]$) requires increased precision. The limited precision of the component hardware operations implies that the error also increases as the input value increases.

Alternative Implementation of Floating-point Sine and Cosine with DW_lp_fp_multifunc

The floating-point sine and cosine operation can also be implemented by DW_lp_fp_multifunc component, which evaluates the value of floating-point sine and cosine with 1 ulp error bound. There will be 1 ulp difference between the value from DW_lp_fp_multifunc and the value from DW_fp_sincos. Performance and area of the synthesis results are different between the DW_fp_sincos and sine and cosine implementation of the DW_lp_fp_multifunc, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between the sine and cosine implementation of DW_lp_fp_multifunc and DW_fp_sincos component, the DW_lp_fp_multifunc provides more choices for the better synthesis results. Below is an example of the Verilog description for the floating-point sine and cosine of the DW_lp_fp_multifunc. For more detailed information, see the [DW_lp_fp_multifunc](#) datasheet.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 24, 0, pi_multiple) U1 (  
    .a(a),  
    .func(func),  
    .rnd(3'h0),  
    .z(z),  
    .status(status)  
);
```

For more information about the floating-point, including status flag bits, and integer and floating-point formats, refer to the [Datapath Floating-Point Overview](#).

Related Topics

- [Datapath Floating-Point Overview](#)
- [DesignWare Building Block IP User Guide](#)

HDL Usage Through Component Instantiation - VHDL

```

library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_fp_sincos_inst is
  generic (
    inst_sig_width : POSITIVE := 23;
    inst_exp_width : POSITIVE := 8;
    inst_ieee_compliance : INTEGER := 0;
    inst_pi_multiple : INTEGER := 1;
    inst_arch : INTEGER := 0;
    inst_err_range : INTEGER := 1
  );
  port (
    inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_sin_cos : in std_logic;
    z_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    status_inst : out std_logic_vector(7 downto 0)
  );
end DW_fp_sincos_inst;

architecture inst of DW_fp_sincos_inst is

begin

  -- Instance of DW_fp_sincos
  U1 : DW_fp_sincos
  generic map (
    sig_width => inst_sig_width,
    exp_width => inst_exp_width,
    ieee_compliance => inst_ieee_compliance,
    pi_multiple => inst_pi_multiple,
    arch => inst_arch,
    err_range => inst_err_range
  )
  port map (
    a => inst_a,
    sin_cos => inst_sin_cos,
    z => z_inst,
    status => status_inst
  );

end inst;

```

```
-- pragma translate_off
configuration DW_fp_sincos_inst_cfg_inst of DW_fp_sincos_inst is
for inst
end for;
end DW_fp_sincos_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_sincos_inst( inst_a, inst_sin_cos, z_inst, status_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 0;
parameter pi_multiple = 1;
parameter arch = 0;
parameter err_range = 1;

input [sig_width+exp_width : 0] inst_a;
input inst_sin_cos;
output [sig_width+exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_sincos
    DW_fp_sincos #(sig_width, exp_width, ieee_compliance, pi_multiple, arch, err_range)
        U1 ( .a(inst_a), .sin_cos(inst_sin_cos), .z(z_inst), .status(status_inst) );

endmodule
```

Revision History

For notes about this release, see the [DesignWare Building Block IP Release Notes](#).

For lists of both known and fixed issues for this component, refer to the [STAR report](#).

For a version of this datasheet with visible change bars, click [here](#).

Date	Release	Updates
October 2020	DWBB_202009.1	<ul style="list-style-type: none"> ■ For STAR 3376758, updated the upper range of <i>sig_width</i> to 32 in Table 1-2 on page 1 ■ For STARs 9001517974 and 9001576897: <ul style="list-style-type: none"> - Updated the description of <i>err_range</i> in Table 1-2 on page 1 - Added new architecture values for <i>arch</i> in Table 1-2 on page 1 - Added “Error Behavior” on page 3 ■ Removed a row of data from Table 1-5 on page 3
January 2020	DWBB_201912.1	<ul style="list-style-type: none"> ■ Corrected port names for DW_lp_fp_multifunc in “Alternative Implementation of Floating-point Sine and Cosine with DW_lp_fp_multifunc” on page 4
July 2019	DWBB_201903.3	<ul style="list-style-type: none"> ■ Removed reference to minPower library in “Alternative Implementation of Floating-point Sine and Cosine with DW_lp_fp_multifunc” on page 4 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

© 2022 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com